30 | 实践篇大串讲：重难点回顾+思考题答疑+知识全景图

2020-10-19 王磊

分布式数据库30讲 进入课程 >

讲述：王磊
时长 16:54 大小 15.48M

你好，我是王磊。

今天这一讲是我们课程的最后一个答疑篇。我会回顾第 24 讲到第 29 讲的主要内容，这部分内容是跳出数据库的架构设计，从应用系统的整体视角展开的。接下来，我照例会集中解答留给大家思考题，同时也会回复一些大家关注的热点内容。

第 24 讲：全球化部署

第 24 讲的主题是全球化部署，更接地气的说法就是“异地多活”。

异地多活的目标是保证在区域级灾难事件的发生时，关键业务仍然能够持续开展。其实，异地多活一直是高可用架构所追求的目标，它的难点是有状态服务的处理，尤其是数据

库。在实践中，有好几种基于单体数据库的方案，但它们都有局限性，无法实现“永不宕机”和“近在咫尺”这两点要求。

分布式数据库基于新的架构设计思想，是有条件达成这两点的。

实现“永不宕机”的前提是让主副本可以在异地机房之间的漂移，这就对全局时钟有更高的要求，必须做到多时间源、多点授时。目前能够支持的产品有 Spanner、CockroachDB 和 YugabyteDB，而采用单点授时的产品是无法支持的，比如 TiDB、OceanBase 以及 PGXC 风格的数据库。

而实现“近在咫尺”则要做到两点：

1. 让主本数据能够主动漂移到用户侧机房，降低写操作的延迟。
2. 使用就近的副本提供读服务，也就是 Follower Read 功能。

关于第二点，目前还没看到完全成熟的方案，难点在于如何保持副本数据的新鲜度，从而避免访问主副本带来的延迟。目前，已经有一些产品尝试通过副本与主本、元数据管理节点间的时间戳同步，降低对主本的依赖，保持副本数据的新鲜度。

最后，分布式数据库的多副本机制在实践中也存在一些缺陷。我们知道 Raft 或者 Paxos 都是多数派协议。按照协议原理，当出现网络分区时，在任何一个联通的子网络中，节点数量必须大于总节点数的一半才能选举出新的 Leader，从而继续对外服务。

那么，在两地三中心或三地五机房模式下，如果机房间网络全部中断，即使主机房节点全部正常也不能提供服务。这是因为考虑到 RPO 为零的目标，主机房的节点通常不会超过半数。这意味着，三机房或五机房的部署方案都会让机房间的网络成为易受到外来攻击的风险点，从而数据库变得更加脆弱。

这一讲的思考题是“课程中提到了 Raft 协议降级处理，它允许数据库在仅保留少数副本的情况下，仍然可以继续对外提供服务。这和标准 Raft 显然是不同的，你觉得应该如何设计这种降级机制呢？”

这个机制并不复杂。首先是固定主副本的位置，只有主机房的副本才能被选举为 Leader，这样就能保证主机房的数据足够新；其次是引入一个“低水位线”的概念，约定节点数量

的下限，允许节点数量少于半数但高于下限时，仍然可以对外提供服务。

事实上，这两点不单是理论探讨。第一点在很多分布式数据库的实际应用中都有实现，因为这种约束本身就能降低运维的复杂度；第二点在 GoldenDB 等数据库中已经可以看到类似的设计。

第 25 讲：逃生通道

第 25 讲的关键词是逃生通道。逃生通道本质上是一个异构高可用方案，这种设计并不常见，但在分布式数据库的应用中是一个普遍需求。背后的原因是，数据库必须实现高可靠，而分布式数据库本身的成熟度不足以让用户安心，需要用更成熟的单体数据库做兜底方案。

高可靠方案的核心是如何将变更信息流推送给单体数据库，这个过程中会面临三个问题。其中性能适配和日志适配这两个问题相对容易处理；第三个问题是实现逃生库的事务一致性，要在变更信息流中添加特殊的时间戳消息，用来标识消息的完整性。

这一讲的是思考题是“对单个数据分片来说 WAL 本身就是有序的，直接开放就可以提供顺序一致的变更消息。而逃生通道作为一个异构高可用方案，往往会使用 Kafka 这样的分布式消息队列。那么单分片的变更消息流在逃生通道中还能保证顺序一致吗？”

答案是不能保证顺序一致，原因在于 Kafka 的分布式架构设计。

Kafka 按照 Topic 组织消息，而为了提升性能，一个 Topic 会包含多个 Partition，分布到不同节点（Broker）。那么，所有发送到 Kafka 的消息，实际上是会按照一定的负载均衡策略发送到不同的 Broker 上。

这样的设计意味着，在 Broker 内部消息是有序的，但多个 Broker 之间消息是无序的。作为消息的消费者，即使接收的只是单个分片的变更消息流，也不能用当前消息的时间戳来判定小于这个时间戳的消息已经发送完毕。

第 26 讲：容器化

在第 26 讲中，我们讨论了分布式数据库进行容器化部署的可行性。其中的核心问题就是 Kubernetes 对有状态服务的支持情况。

StatefulSet 是有状态服务的核心对象，我们围绕着它介绍了对应持久卷（Persistent Volume）特性的变化。随着 Kubernetes1.14 版的发布，基于本地磁盘的本地持久卷（Local Persistent Volume）已经达到生产级要求，有望彻底解决数据库容器化部署最大的障碍，I/O 性能问题。

分布式数据库本身服务的复杂性也是不容忽视的问题。通过 Kubernetes 自带的简单协调机制很难定义产品级的管理策略。所以，Kubernetes 提供了 Operator 这种扩展机制，让每个产品可以自行定义控制策略。目前，有不少分布式系统开发了自己的 Operator。

这一讲的思考题是“ 资源调度是 Kubernetes 的一项核心功能，那么除了 Kubernetes 你还知道哪些集群资源调度系统？它们在设计上有什么差异吗？”

随着很多基础系统向分布式架构过渡，集群资源调度成为一个非常广泛的需求，尤其是数据密集型系统。除 Kubernetes 外，知名度最高的可能就是 Yarn 和 Mesos，它们都是大数据生态体系下的工具。

Yarn 是 Hadoop 体系下的核心组件，被称为“数据操作系统”，所调度的资源单位和 Docker 有些相似，比如也使用了 Cgroup 隔离资源。Mesos 更是直接从大数据领域杀入容器编排领域，成为 Kubernetes 的直接竞争对手。

第 27 讲：产品测试

第 27 讲中，我们谈的核心内容是测试。测试是保证软件质量的重要手段，而对于分布式数据库来说测试的难度更大，也更为重要。我们先后介绍了 TPC-C、Jepsen 和混沌工程三种测试方式。

TPC-C 是由国际事务性能委员会发布的、针对 OLTP 数据库的测试规范，已经被业界广泛接受。

对于分布式数据库来说，是否正确实现了数据和事务的一致性也是测试的关键。Jepsen 是针对这类测试的权威工具，很多知名的分布式存储系统购买了 Jepsen 的测试服务。

混沌工程和 Jepsen 一样，都是通过注入底层故障的方式进行测试。但混沌工程的测试对象更广泛，涵盖了所有的分布式系统。同时，混沌工程是一种不同的测试文化，强调在生产环境注入故障，在受控的范围内观测系统，体现了反脆弱的思想。

最后，所有的测试都是在证伪，只能发现错误，不能证明正确性。而形式化验证弥补了这个缺憾，提供了证明的方法。在软件工程领域，Lamport 发明的 TLA，衍生出一系列方法、工具甚至编程语言，使我们可以用更严谨的方式确认逻辑的正确性。

这一讲的思考题是“除了分布式数据库，对于其他类型的分布式存储系统，你知道有哪些主流的测试工具吗？”

由于分布式系统的复杂性，测试工具就显得更加重要。除数据库外，比较典型是针对分布式键值系统的测试工具，例如 YCSB。YCSB（Yahoo! Cloud Serving Benchmark）是一个开源的测试框架，你可以从 Github 上下载到源码。

YCSB 支持几乎所有的主流分布式键值系统，包含 HBase、Redis、Cassandra 和 BigTable 等。YCSB 支持典型的 PUT\GET\SCAN 操作，你可以很容易地在它的基础上扩展，增加对其他键值系统的支持。

第 28 讲：选型案例

在第 28 讲，我们讨论的话题是如何进行分布式数据库产品选型。因为对于数据库产品来说，金融场景是公认的试金石，所以我们这一讲主要介绍了银行业的选型情况。

总的来说，银行的态度分为三种。

最保守的方案是采用分库分表，甚至单元化的方案，在应用层付出更大的代价，而在数据库层沿用单体数据库，保持充分的稳定性。这种方式的前提是必须有足够的业务驱动力，才能推动应用系统进行重大的调整。

第二种方式是自研数据库，周期长、投入大，但可以得到更加切合自身的解决方案。PGXC 数据库的难度较低，所以推进更快。

第三种方式是采购分布式数据库。目前随着技术的成熟，这种模式越来越普遍，甚至一些业务量较小的城商行也上线分布式数据库。

这一讲的思考题是“产品选型是系统建设中非常重要的工作，你觉得针对一个新建系统做产品选型时还要考虑哪些因素呢？这个产品，并不限于数据库？”

对于分布式数据库的选型，我认为最重要的是分析业务场景对于事务和查询方面的要求。事实上，这两部分内容也占据了我们课程的大半。在实践中，有的海量并发业务其实并没有相关性，也就是说，不同事务间数据重叠的概率非常低，这时就可以考虑分库分表方案。不过，多数的分库分表方案只能对数据做单向的路由分发，如果有跨分片的查询需求则很难满足。虽然，也可以通过 CDC 等工具导出到分析型系统实现这一点，但这时就要平衡系统的整体复杂度了。

跳出具体技术，我认为产品选型要结合具体项目目标，选择尽量少的产品或技术组件，尽量不采用同质化的多种技术，让架构更加简洁和优雅。

第 29 讲：产品图鉴

第 29 讲，我们转换到了产品视角，对课程内容做了一次完整的回顾。

我相信，很多同学都会有这样的问题：哪些分布式数据库更有发展，值得花时间学习？

这一讲，我根据个人经验列举了常见的十款数据库产品，并给出了相关内容在课程中对应的位置，方便你快速查找。

谈到产品，我们还是依据第 4 讲的分析框架将产品分为三类，NewSQL、PGXC 和其他。其中 NewSQL 是我们课程的重点，原因在于它体现了学术界和工业界近年来的很多创新。我们以 Spanner 和 F1 为基础，进行了比较完整的说明，并在此基础上介绍了其他四种 NewSQL 数据库的特点。

作为另一个主要流派，PGXC 是从分库分表方案演进而来的，演进过程的里程碑就是全局时钟。因为 PGXC 更加依赖单体数据库，技术门槛相对较低，所以产品数量就较多，很难全部列举。最终，我们只挑选了比较典型的三款产品。PGXC 虽然在架构上更加稳健，但部分单体数据库的许可证要求基于其开发的软件也必须开源，所以对于没有按照这一要求执行的商业产品来说，始终存在着法律风险。

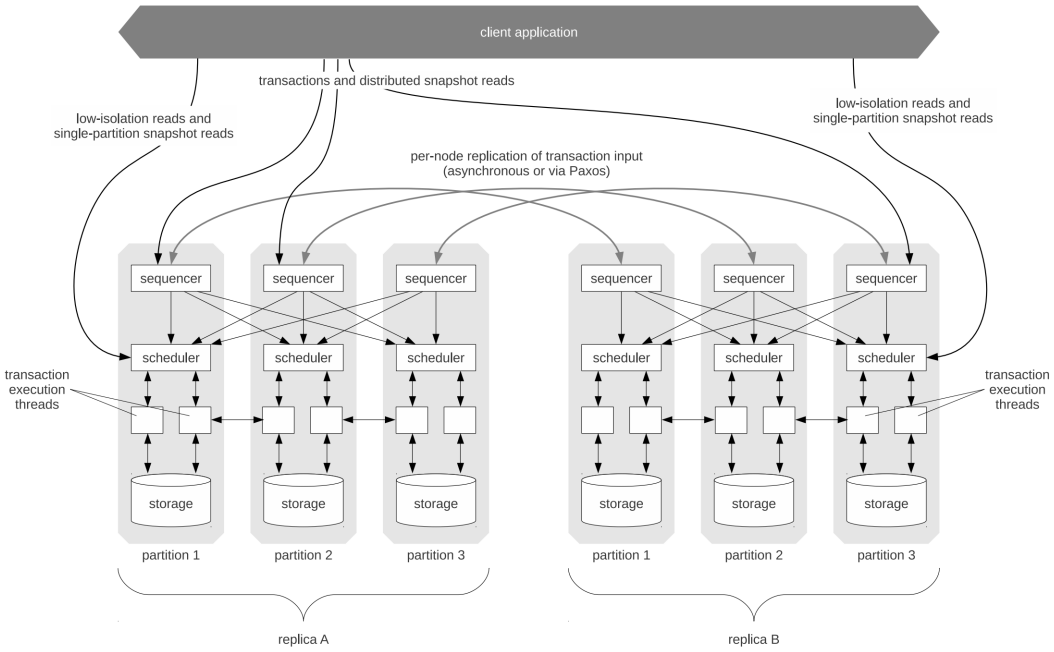
最后，作为小众产品的代表，我们简述了 VoltDB 和巨杉数据库。它们也都是各种产品评测中的常客，但是由于各种原因，应用案例会相对少些。在一些特定场景下，这些产品也是重要的备选项。

这一讲的思考题是“除了第 29 讲提到的数据库之外，你还了解哪些有特点的分布式数据库吗？”

我在课程中还提到过两种不同的分布式数据库，分别是 FoundationDB 和 FaunaDB。最早，

FoundationDB 是一个开源项目，而后被苹果收购后闭源，2018 年又重新开源。严格来说，它是一个支持 ACID 语义的 NoSQL 数据库，有比较独特的乐观并发控制。FoundationDB 可以通过上层的扩展可以支持更多的场景，而在增加了 Record Layer 后，它就成为一个关系型数据库，不过还不支持 SQL 接口。总的来说，FoundationDB 值得我们持续关注。

对于 FaunaDB，可能了解的人更少些。它是一款商业数据库，从 NoSQL 逐渐过渡成为分布式数据库。FaunaDB 的理论原型是 Calvin，它和 Spanner 一样在 2012 年发表了原型论文，被 SIGMOD 收录。虽然它们的目标都是提供强一致性的分布式关系型数据库，但设计思路截然不同。



在 Calvin 的架构中设计了 Sequencer 和 Scheduler 两个组件。

其中 Sequencer 是基于锁机制为输入的事务操作进行排序，这样在事务真正执行前就能获得确定性的顺序，避免了过程中的无效交互和资源冲突。Scheduler 则是按照这个顺序执行，调度不同的事务执行线程完成最终操作。

这个设计思路被称为确定性并发控制，对应的数据库则称为确定性数据库（Deterministic Database）。

确定性并发控制算法的基本思路是，确保不同的副本始终能够获得相同的输入，所以就能独立地产生相同的结果，从而避免使用开销更大的提交协议和复制协议。通常，这类算法可以降低协调工作的通信开销，提高分布式事务的处理效率。而它的代价之一是无法为客户端提供一个交互操作过程，这就限制了它的使用场景。

目前基于 Calvin 理论模型的工业产品，似乎只有 FaunaDB。作为一种有代表性的分布式数据库，它值得我们继续关注。

此外，确定性数据库也是学术领域的重要研究方向。VLDB2020 收录的一篇论文“Aria: A Fast and Practical Deterministic OLTP Database”，就探讨了对确定性控制算法的优化方案，如果你有兴趣可以仔细研究下。

小结

最后，我想把这 30 讲的组织方式总结一下。

在基础篇，我重点铺垫了数据库和分布式架构的基本概念，这些内容可以独立成篇，阅读难度相对较低。

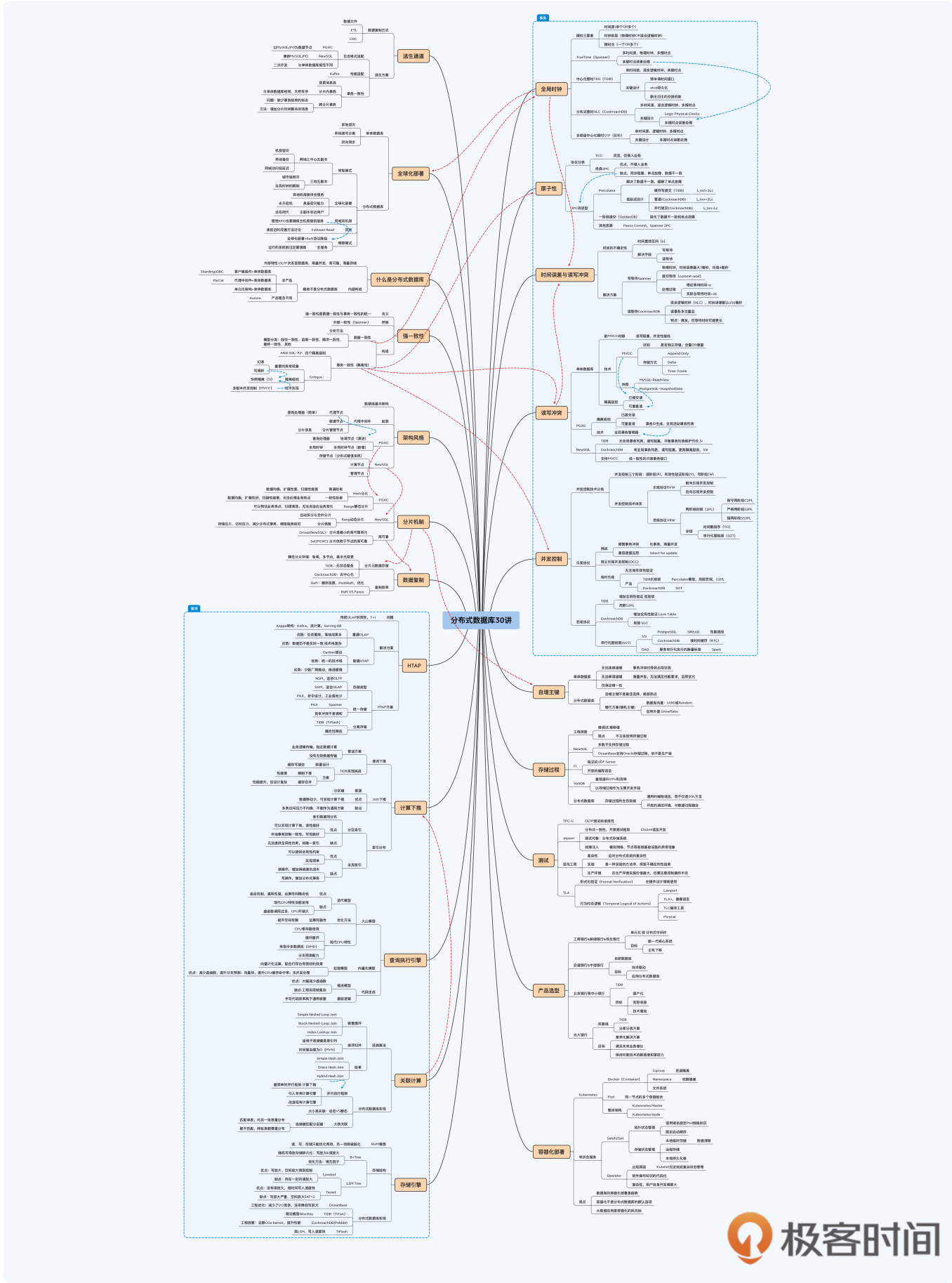
在开发篇，我们一起深入到分布式数据库的内部，对关键设计进行拆解，这一部分涉及大量的学术论文和工程实践。其中，事务和查询无疑是两座高峰，为了你有个歇脚喘气的地方，我在中间也穿插了一些相对简单的内容。

最后的实践篇，其实已经跳出数据库的范畴。这一部分的话题主要是围绕应用系统建设的整体视角展开，目的是希望将课程与你的日常工作衔接起来。

对于分布式数据库来说，30 讲的篇幅还是很短的，有些原理可能还没有完全说透，还有些有趣的东西或许被遗漏了。不过，这并不要紧，因为课程虽然结束了，但我们的交流还在继续。

如果你对课程任何内容有疑问，欢迎在评论区留言和我一起讨论。要是你身边的朋友也对分布式数据库感兴趣，你也可以把这门课程分享给他，我们一起讨论。

分布式数据全景图 4/4



学习资料

Alexander Thomson et al.: *Calvin: Fast Distributed Transactions for Partitioned Database Systems*

Yi Lu et al.: *Aria: A Fast and Practical Deterministic OLTP Database*

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 29 | 产品图鉴：哪些分布式数据库值得看?

下一篇 结束语 | 享受职业带给你的快乐

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。